

INTRODUKTION

i kørsel af algolprogrammer på GIER samt en gennemgang af de vigtigste hjælpeprogrammer.

Vi antager, at man har hullet sit algol program, og at strimlen har følgende udseende:

```
begin
.
.
<program>
.
end
```

Når man ønsker at køre dette program, sættes strimlen i læseren, og der trykkes på reset.

algol
se hjælp 3,
side 25

Algoloversætteren kaldes ved at skrive på skrivemaskinen

l, algol, 1 <

Når det er gjort, går læseren igang med at 'spise' strimlen, samtidig med at programmet udskrives med nummerring af linierne på linieskriveren.

syntaktiske fejl
se alg. man.
s. 80-84

Nu kan det jo være, at der er syntaktiske fejl i programmet, og i så fald kan det ikke oversættes, i stedet leverer oversætteren en liste over de fejl, der er fundet, samt en angivelse af, hvilke linier fejlene forekommer i. Hvis der er fejl, sluttet af med udskriften sorry (med rødt) på skrivemaskinen. Det hænder også af og til, at der ikke er syntaktiske fejl i et program, i så fald kommer udskriften ok på skrivemaskinen (med sort). Programmet er så oversat, og det oversatte program befinder sig på et område på tromlen, der hedder work.

oversætt. ok

work

run

Når programmet er ok og oversat, og man gerne vil have det kørt, sættes en evt. datastrimmel i læseren, og der trykkes på læserens reset. På skrivemaskinen taster:

run <

runtime-error
se alg. man.
s. 78-79

run er et hjælpeprogram, der aktiverer det oversatte program, der befinder sig på work. Men der kan jo stadig være fejl i programmet, og fejlene kan være af så alvorlig karakter, at kørslen afbrydes, dette sker med en fejludskrift på skrivemaskinen med angivelse af linienumre.

hp-knap

Der kan også være fejl i et program, der bevirker, at programmet leverer forkerte resultater eller f. eks. aldrig bliver færdigt, og så kan det blive nødvendigt at stoppe programmet. Dette gøres ved at trykke på hp-knappen, som stopper programmet, så snart gier er færdig med at udføre den maskinordre, den var i gang med.

Fejludskrifter fra hjælp Når man benytter et hjælpeprogram forkert, kan der forekomme forskellige udskrifter. Disse kan findes i hjælp 3, s.15.

Før vi fortsætter gennemgangen af de mest brugte hjælpeprogrammer, vil vi lige fortælle lidt om input/outputmedier samt fortælle, hvordan maskinen opfatter hjælp 3 information.

input/output Når man ankommer til gier, vil den normale situation være, at gier venter på input fra skrivemaskinen (dette kan ses af, at den grønne lampe lyser). Man siger, at skrivemaskinen er løbende input. Skrivemaskinen har navnet `t` (type), skriver man `r<`, har man ændret løbende input fra at være skrivemaskinen til at være læseren, og gier venter nu på input fra læseren, og den grønne lampe slukkes, læseren har altså navnet `r` (reader).
 Af outputmedier findes følgende:
`l` linieskriveren, som har navnet `l` (lineprinter)
`p` perforatoren, som har navnet `p`
`w` skrivemaskinen (som outputmedium), der har navnet `w` (writer).

parameterliste For at få oversat vores algolprogram, benytter vi os af kaldet:

`l, algol, 1 <`

Dette kaldes hjælp 3 information, de enkelte elementer i listen adskilles af komma, og listen afsluttes med `<` (hak). Efter indlæsningen, der slutter, når der mødes et hak, vil hjælp ud fra kataloget identificere listens navne et efter et:

Hvis navnet er et navn på et outputmedium, vælges dette som hjælps outputmedium, og hjælp fortsætter i listen.

Hvis navnet er en inputenhed eller et tekstområde på baggrundslageret, vælges dette som hjælps løbende inputmedium, og hjælp fortsætter i listen.

Hvis navnet er et hjælpeprogram, vil resten af listen blive overflyttet til en særlig kanal på tromlen (parametersporet), og hjælp overgiver kontrollen til det pågældende hjælpeprogram, som så tager sig af resten af parametrene.

Hvis navnet ikke findes i kataloget, kommer udskriften `undef` (med rødt) på skrivemaskinen, og alt er glemt.

I ovenstående eksempel vil `l`, som er navnet på linieskriveren, bevirke, at denne vælges som outputmedium, `algol` er et hjælpeprogram, der sætter oversættelsen i gang, og `1` er en parameter til `algol`, der bevirker, at vi får en udskrift af hver linie af programmet. Hvis vi i stedet havde skrevet f.eks. `7`, ville vi have fået en udskrift af hver syvend linie.

Retning af programmer og tekster

edit
se hjælp 3,
s. 32

Hvis man under oversættelse eller kørsel af et program finder fejl, må man naturligvis rette fejlene. En nærliggende løsning er at bære sin strimmel ind til en flexo-writer og så rette strimlen der. Dette kan gå, når programmerne eller strimlerne er små, men ved større programmer er det en meget besværlig løsning, derfor findes der et hjælpeprogram, som er beregnet til retning af programmer og tekster. Dette program hedder edit. Det er klart, at hvis vi vil rette pr. automatik, må vi have en liste med alle de rettelser, vi ønsker at lave. Rettelserne skal stå i samme rækkefølge som fejlene forekommer i, retningen foregår nu således:

Først læses rettelsesstrimlen fra hjælps løbende inputmedium, og på skrivemaskinen udskrives, hvor mange rettelser der er.

Dernæst rettes teksten, som befinder sig på et nærmere specificeret inputareal, og den rettede tekst lægges i samme ombæring ud på et nærmere specificeret outputareal.

Til sidst udskrives hvor mange rettelser, der blev gennemført. De to tal, der nu er udskrevet, skulle gerne være ens ellers er der fejl i rettelsesstrimlen.

Lad os antage, at vi har et program på strimmel, der skal rettes, og en strimmel med rettelser. Det rettede program vil vi gerne have ud på lineskriveren, så skriver vi:

$r, \text{edit}, \underline{i}r, \underline{o}l <$

r er navnet på læseren, den er derfor nu valgt som løbende input (jfr. afsnittet om parameterliste). edit er navnet på et hjælpeprogram, som nu overtager kontrollen. De to sidste par parametre er parametre til edit og betyder:

\underline{i} (inputareal r (læser))
 \underline{o} (outputareal) l (lineskriver)

altså:

Rettelsesstrimlen læses først fra læseren, den tekst, der skal rettes, sættes dernæst i læseren, og den rettede tekst udskrives på printerens.

Vi kunne have gjort kørselen lidt mere hensynsfuld overfor de øvrige brugere, nemlig ved forrest på rettelsesstrimlen at skrive:

$\text{edit}, \underline{i}r, \underline{o}l <$

og så på skrivemaskinen blot have skrevet $r <$.

Virkningen er ganske den samme, thi ved at skrive $r <$ ændrer vi hjælps løbende input til at være læseren. Det næste, der så læses, er ovenstående hjælp 3 kald, og vi er nu tilbage i den foregående situation.

Fordelen ved at have sine hjælp 3 kald stående på en strimmel er for det første den, at man kan sidde i ro og mag og tilrettelægge sin kørsel ved en flexowriter, og for det andet at læseren læser med en hastighed af ca. 2000 tegn/sek. (eller sagt på en anden måde ca. 1000 gange så hurtigt som en bruger.)

Øvelse:

På rettellesstrimlen står:

edit, ir, op <

På skrivemaskinen skrives: $r <$
Hvad sker? (Løsning andetsteds på siden)

Ved at skrive $r <$ har man valgt læseren som løbende inputmedium - hjælp læser kaldet edit, ir, op, fra læseren, kontrolle overgives til edit - derefter rettelleslisten, som sidder i læseren, når den er læst, skrives antallet af rettelser, nu sættes strimlen med teksten, der skal rettes, i læseren, der tages et space på skrivemaskinen, og samtidig med at teksten læses, skrives den rettede tekst på perforatoren. Til sidst udskrives antallet af fundne rettelser.

Rettelseslisten Rettelseslisten har form som en liste af enkeltrettelser, som skal komme i den rækkefølge, hvori fejlene forekommer i programmet. En enkeltrettelse består af tekststrengene adskilt af (understreget punktum):

<kopier til og med> <indsæt> <slet til og med>

Når listen er færdig afsluttes med stopcode eller å. Det bemærkes, at space og cr er blinde i <kopier til og med> og <slet til og med>.

```
programstrimmel: begin integer i, j, x;
                  for k:=1 step 1 until 10 do
                  ...
                  writetekst (< Jens Olsens program >);
                  ...
                  end t<
                  <stopcode>
rettelsesstrimmel: edit, ir, ol
                   i, j, k.x.
                   :=1 step p.
                   <.<.
                   <stopcode>
```

Rettelsen foregår ved at lægge rettelsesstrimlen i læseren, trykke på læserens sæt og taste r< på skrivemaskinen. Når rettelseslisten er læst, udskrives på skrivemaskinen 3 (antallet af rettelser.), derefter sættes programmet i læseren, der taster et space og det rettede program kommer ud på printerens, samtidig med at strimlen læses.

Baggrunds-
lageret

jfr. algol

Før vi fortsætter med at fortælle om edit, vil vi forklare lidt om baggrundslageret på gier. Man kan reservere dele af baggrundslageret til opbevaring af programmer og data. Et sådant reserveret område skal have et navn, som brugeren selv bestemmer, blot må to forskellige områder ikke have samme navn. Navnet er en sædvanlig identifier med den begrænsning, at der kun må benyttes små bogstaver. Navnet, tillige med en beskrivelse af områdets placering på baggrundslageret, opbevares i kataloget. Alt hvad der ikke er reserveret har navnet free og kan frit benyttes af alle brugere. Navne på spdanne områder kan specificeres som input-outputområder til edit.

edit, i free, opax <

tager den tekst, der skal rettes fra free og afleverer den rettede tekst på pax. Rettelseslisten læses som sædvanlig fra løbende input.

free, edit, i carlo, o peter <

Rettelseslisten læses fra free, den tekst der skal rettes læses fra carlo, og den rettede tekst udskrives på peter.

Hvis rettelseslisten er tom, vil der foregå en simpel kopiering af input teksten.
Kaldet:

```
edit, _ir, _ofree <
```

med en tom rettelsesliste vil således bevirke, at den tekst, der står på strimmel, vil blive kopieret til det frie område. Når teksten befinder sig på free kan den reserveres med hjælpeprogrammet `res` ved kaldet:

`res`
se hjælp 3,
s. 40

```
res, frederik <
```

Programmet befinder sig nu på baggrundslageret under navnet `frederik`. Et område, der er reserveret på denne måde, kan påregnes at være tilgængeligt resten af dagen. Vil man gerne have programmet stående længere tid, kan man bruge reservering med dato:

```
res, _d 24.12.72, frederik <
```

Datoreservering sker altså med parameteren `_d` efterfulgt af den sidste dato, man ønsker at kunne benytte sit område. `frederik` vil altså ikke blive fjernet før 25.12.72. Normalt reserverer man højst to uger frem ad gangen, hvis man, når de to uger er gået, mener at have brug for sit område længere tid, kan man ændre datoen med hjælpeprogrammet `redate`:

`redate`

```
redate, _d 1.10.72, frederik <
```

vil således ændre sidste dag til den 1.10.72 (uanset om den tidligere dato er før eller efter den nye dato). Hvis man på et eller andet tidspunkt ikke længere har brug for sit område, skriver man:

```
redate, frederik <
```

Dette svarer til:

```
redate, _d 0.0.0, frederik <
```

Området vil da normalt være væk den næste dag.

NB! Man kan kun bruge `redate`, hvis området oprindeligt er reserveret med dato.

`move`
se hjælp 3,
s. 36

Til slut vil vi lige omtale endnu et vigtigt hjælpeprogram, nemlig programmet `move`, der benyttes til kopiering af et område ind i et andet område.

```
move, pax1, pax2 <
```

flytter informationen i `pax1` til `pax2`, således at man nu har den samme information på begge områder.

`booked`
se hjælp 3,
s. 50

Når man flytter information til `free`, enten ved hjælp af `edit` eller `move`, vil hjælp altid vide, hvor meget af `free` der er brugt (`booked`), og når man så bruger `res`, vil `res` vide, hvor meget der er brugt af `free` og reservere dette.

En kortfattet vejledning i brug af algol, edit, run.

Rettestrimmel: edit, ir, ofree <
 <rettelsesliste>
 <stopcode>

programstrimmel: l, algol, l <
 <program>
 t <
 <stopcode>

efterflg. rettestrimmel: edit, i free, o free <
 <rettelsesliste>
 <stopcode>

Indkøringsproceduren er:

Man tager en udskrift af sit program både på print og på strimmel. Alle rettelser påføres udskriften, derved kan man altid se, hvilke rettelser man har lavet i tidens løb. Rettelser sker ved at modificere sin rettelsesstrimmel. Når programmet er indkørt, tager man en ny udskrift (på print og strimmel) af sit program. Hvis man har mange rettelser, kan man forny udskrift og strimmel, hver gang man har afsluttet en del af indkøringen.

1. første rettestrimmel sættes i læseren
2. tast r <
3. udskrift på skrivemaskine af antal rettelser
4. sæt programstrimmel i læseren
5. tast et space på skrivemaskinen
6. udskrift på skrivemaskinen af antal fundne rettelser. Hvis de to tal ikke stemmer overens, så find fejlen på rettelsesstrimlen.
7. tast free <. Programmet bliver så oversat samtidig med at man får en linieudskrift (p.g.a. l, algol, l < forrest på strimlen)
8. hvis der er fejl i programmet, så find fejlen(e), påfør dem udskriften, modificer rettelsesstrimlen og gå til pkt. 1.
9. hvis programmet er ok, testes run < - og programmet kører nu
10. når programmet er fejlfrit og gennemtestet, udtages sidste version af programmet ved hjælp af

edit, i free, op <
 <stopcode> ell. å

Fred Mosekjær Madsen